XILOG

VOL 1 NO 9

The Microcomputer Magazine for VZ200, VZ300 and AQUARIUS users.

Published monthly by the VZ/AQUARIUS Users Group and
Distributed free nationally by Dick Smith Electronics (Newmarket).

IMPROVING HI RES GRAPHICS

The VZ is pretty limited in its graphics commands. SET, RESET and POINT are the only ones available and a few more commands would certainly not go amiss.

MOVE and DRAW are two commands available on a number of machines for which a lot of programs are already written and may be used on the VZ with very little effort.

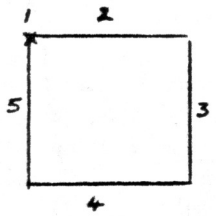
MOVE is equivalent to -
- HPLOT- Apple
- PLOT- Atari
- MOVE- BBC

DRAW is equivalent to-
- HPLOTTO- Apple
- DRAWTO- Apple
- DRAW- BBC

MOVE merely sets the point at which a line drawn on the screen will begin, whilst DRAW determines the end point of the line and actually draws the line. If subsequent DRAW commands are issued a line is drawn from the position of the previous DRAW command to that of the last DRAW command.

Thus to draw for example a square on the screen the commands would be typically:-

```
10 MOVE 20,20
    (go to position 1)
20 DRAW 30,20
    (draw line 2)
30 DRAW 30,30
    (draw line 3)
40 DRAW 20,30
    (draw line 4)
50 Draw 20,20
    (draw line 5 )
```



To mimick these commands on the VZ a small subroutine is needed which is listed below. It draws a line between two x,y coordinates - A,B and C,D.

```
50000 IFA<CTHENX1=A:Y1=B
   :X2=C:Y2=D:GOTO50020
50010 X1=C:Y1=D:X2=C:X2=
   A:Y2=B
50020 XX=X2-X1
50030 IFXX=0THENSL=1000E
   LSESL=(Y2-Y1)/(X2-X1)
50040 IFABS(SL)>1THENGOT
   O50130
50050 SET(X1,Y1)
50060 FORI=1TOX2-X1-1
50070 TX=X1+I
50080 TY=Y1+INT(I*SL)
50090 SET(TX,TY)
50100 NEXTI
50110 SET(X2,Y2)
50120 GOTO50190
50130 IFB<DTHENX1=A:Y1=B
   :X2=C:Y2=DELSEX1=C:Y1
   =D:X2=A:Y2=B
50140 FORI=1TOY2-Y1-1
50150 TY=Y1+I
50160 TX=X1+INT(I/SL)
50170 SET(TX,TY)
50180 NEXTI
50190 A=C:B=D
50200 RETURN
```


Material intended for "Xilog" should be submitted either on Tape (preferably several copies) or be Typed and be accompanied by adequate explanatory notes including senders name and address. Material submitted to "Xilog" becomes the property of the VZ/AQUARIUS Users Group and may be treated at the editors discretion.
Contributions without reward to - Xilog, 22 Ussher Place, Pakuranga, Auckland. ph AK 562-166

To use just enter the subroutine (and save it to tape for further use) then substitute-
A=X: B=Y
for MOVEx,y and

C=X: D=Y: GOSUB 50000
for DRAW x,y.

The program to draw a square then reads-

```
2 MODE(1)
10 A=20:B=20
20 C=30:D=20:GOSUB50000
30 C=30:D=30:GOSUB50000
40 C=20:D=30:GOSUB50000
42 C=20:D=20:GOSUB50000
50 GOTO50
```

The subroutine is able to draw lines of any angle, including vertical, and packs the dots in the best format i.e. like this-

```
x
 x
  x
   x
    x
```

and not like this-

```
x

 x

  x
```

SORTING BY

GRAEME ROSS AUCKLAND

There are many different types of general purpose sorts available to day like
- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Heap sort
- Quick sort

Regardless of which sorting routine is used, the time it takes to sort a table increases exponential to the increase in the length of the table. That is to say that if you double the length of the table it will take four times as long to sort.

THE QUICK SORT:-

The Quick sort chooses at random any element in the table and sorts all the numbers each side of this table element. It moves the large numbers to one side and the small numbers to the other side of this element. This type of sort is good when some of the table may already be sorted, however if the table is completely random sequence than a Quick sort is no faster than a Bubble sort.

THE INSERTION SORT:-

This sort routine chooses the second element in the table and places it in sequence on the left of the table and moves all the other elements to the right. Then it chooses the third element and does the same, then uses the fourth...etc.

THE BUBBLE SORT:-

This is a very common and is easy to program sort, it compares two elements of the table and swaps them if they are not in the correct sequence.

Below is an example of the a Bubble sort of a table with four elements.

```
<-----pass one-----><--pass two--><-pass three>
 ---    ---    ---    ---    ---    ---    ---
! 4 !  ! 3<!  ! 3 !  ! 3 !  ! 2<!  ! 2 !  ! 1<!
! 3 !  ! 4<!  ! 2<!  ! 2 !  ! 3<!  ! 1<!  ! 2<!
! 2 !  ! 2 !  ! 4<!  ! 1<!  ! 1 !  ! 3<!  ! 3 !
! 1 !  ! 1 !  ! 1 !  ! 4<!  ! 4 !  ! 4 !  ! 4 !
 ---    ---    ---    ---    ---    ---    ---
```

'<' indicate the elements of the table tha
that have been swapped.

Each time you you pass through the table one less comparison is required since the lowest/highest element will already be at the top/bottom of the table depending what type of sort you are doing (ascending or descending).

Below is a small program for a Bubble sort

```
10 REM BUBBLE SORT (ASCENDING)
20 REM BY GRAEME ROSS
30 REM
40 REM  TL = TABLE LENGTH
50 REM  TBL= TABLE NAME
60 FOR X = 1 TO TL
70   FOR Y = 1 TO TL - X
75  REM SWAP ELEMENTS IF THE NOT IN SEQUENCE
80   IF TBL(Y) < TBL(Y+1) THEN TEMP=TBL(Y)
     TBL(Y)=TBL(Y+1):TBL(Y+1)=TEMP
90   NEXT Y
100 NEXT X
110 REM
120 REM SORT END
130 REM*************************************
```

NOTE:- (1) To the above sort you could add a flag to see if there was any swapping of variables. If no swaps had occurred then the table is now sorted (this prevents the routine from sorting an already sorted table).

(2) If you wish to few the table as it is been sorted add the following lines:-

```
81 FOR Z = 1 TO TL
83 PRINT TBL(Z)

85 NEXT Z
87 INPUT A$ "PRESS <RETURN> TO CONT ";A$
89 ?"------------------------------"
```

(3) These routines also work for string tables i.e TBL$(x).

MACHINE CODE BASICS

The method of assembling a machine code program, so far described: requires you to look up a table so that you can POKE a number representing the op code into memory. For obvious reasons this method is called "Hand Assembly". For the beginner it is an excellent way to become familiar with machine code since the inevitable mistakes and the subsequent crashes they cause result only in a temporary set-back - assuming that the program was saved on tape BEFORE executing.

However, looking up tables can rapidly become a bore. A chore which the computer is better equiped to do and do fast. A program to enter machine code in symbolic form and output ready to run compiled code already exists for the VZ and is known as EDITOR/ASSEMBLER or EDASM. (a Dick Smith product) Not only does EDASM allow you to directly type in such statements as :- LD BC,64 or LD HL,12 and RET, etc; but a range of useful functions such as insert, delete, save to tape, and merge two programs together makes inputing machine code easy.

Only one snag; every attempt to run a dud program will still result in a crash and that means making another cup of tea (or coffee) while the EDASM program AND the MACHINE CODE source programs are reloaded in order that you can try again.

EDASM can create two forms of output- Source and Object.

Source is the symbolic form of the program- the LD BC, RET, etc; while the Object is the actual program that runs in memory.

EDASM allows machine code to be put and run anywhere in memory and as long as the place it's put has been reserved by some means or another all will be well. (Use the high resolution graphics memory as described in last months issue and you can ignore that problem)

Now comes the best part:- when you start writing programs it will become evident that you will often need the program to jump all over the place and to read and manipulate data stored in various locations.

This necessitates keeping track of where things are and involves even more boring work. eg L. BC,(nn)- load the BC register pair with the value stored at locations nn and nn+1 or JP,(nn)- jump to location nn and execute the instruction stored there.

LABELS are the answer

The use of labels allows you to forget about where instructions and data are stored. Labels simply mark the 'spot' or location to which some other part of the program may refer. The actual locations are figured out by the computer and those values substituted into the final object code on assembly.

All this is by way of an introduction to the "Write a Message" program as promised in last months Xilog. By using EDASM you simply type in each line as written, assemble it and run it. As a beginner you probably haven't got a copy of EDASM, in which case you will be hand assembling and need to substitute real location values for the labels.

Listed below is the same program expressed in various ways which will hopefully make it all clear.

EDITOR SOURCE LISTING

```
001 ;MESSAGE OUTPUT PROGRAM     COMMENT LINE - LIKE REM
002        LD    HL,MSG         LOAD HL REGISTER WITH START LOCATION OF MESSAGE
003        CALL  28A7H          CALL UP THE PRINT ROUTINE
004        RET                  RETURN TO BASIC
005 MSG    EQU   $              MARKS LABEL-MSG AS BEING AT CURRENT MEM LOCATION DENOTED AS $
006 *READY ETC MESSAGE*         MESSAGE STRING
007        DEFB 0DH             CARRIAGE RETURN
008        DEFB 0               NULL BYTE TO TERMINATE
```

EDITOR SOURCE LISTING WITH OBJECT CODE EQUIVALENT AFTER EACH LINE.

```
001 ;MESSAGE OUTPUT PROGRAM
002       LD   HL,MSG
7200 21 07 72
003       CALL 28A7H
7203 CD A7 28
004       RET
7206 C9
005 MSG   EQU  $
006 *READY ETC MESSAGE*
7207 READY ETC MESSAGE
007       DEFB 0DH
7218 0D
008       DEFB 0
7219 00
ERRORS : 00000
BYTES FREE :- 11442
```

REAL ECONOMY!!

This little program acts as a mini word processor. It can even output UPPER and lower case to a printer by entering inverse characters for lower case.

```
10 INPUTA$
20 A=LEN(A$)
30 FORI=1TOLEN(A$)
50 B$=MID$(A$,I,1)
70 IFASC(B$)>192THENB$=CHR$(ASC(B$)-96)
80 IFASC(B$)=128THENB$=" "
90 A$=LEFT$(A$,I-1)+B$+RIGHT$(A$,A-I)
100 NEXT
110 LPRINTA$
120 GOTO10
```

HEX DUMP

```
7200 21 07 72
7203 CD A7 28
7206 C9
7207 52 45 41 44 59 20 45 54 43 20 4D 45 53 53 41 47 45
7218 0D
7219 00
```

SAME PROGRAM WRITTEN IN BASIC

```
100 X=29184:Y=30862
200 READ A:IFA=500THEN600 ELSEPOKE X,A:X=X+1:GOTO200
600 POKE Y,0:POKE Y+1,114
1000 DATA 33,7,114,205,167,40,201,82,69,65,68,89,32,69,84,67,32
1001 DATA 77,69,83,83,65,71,69,13,00,500
1100 X=USR(U)
```

STILL WANTED

ARTICLES FOR XILOG
ARTICLES FOR XILOG
ARTICLES FOR XILOG
ARTICLES FOR XILOG